bdiGDB

BDM debug interface for GNU Debugger

CPU32 / CPU32+

User Manual

Manual Version 1.05 for BDI2000

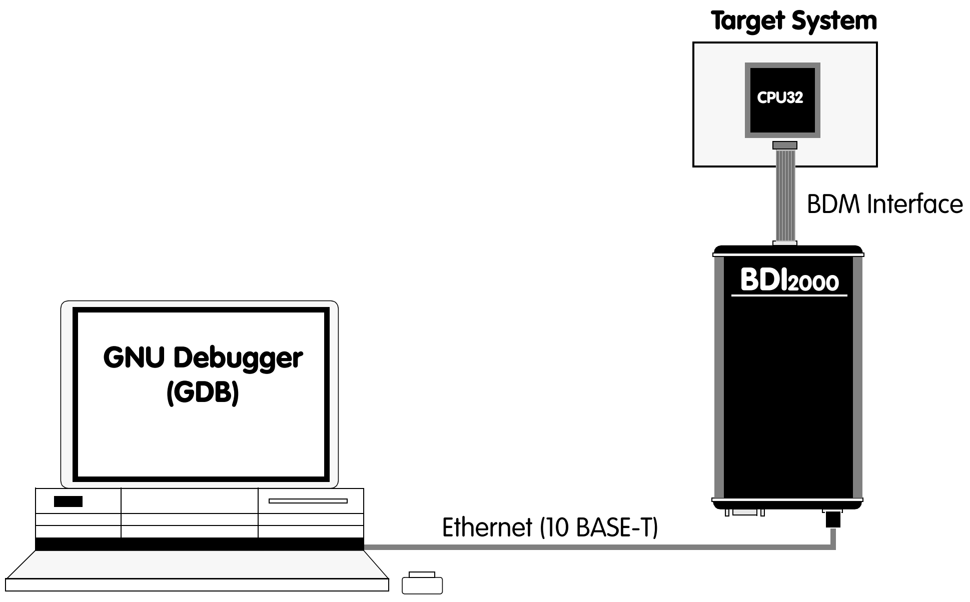
abatron

1 Introduction

bdiGDB enhances the GNU debugger (GDB), with Background Debug Mode (BDM) debugging for CPU32/32+ based targets. With the built-in Ethernet interface you get a very fast code download speed of up to 150Kbytes/sec. No target communication channel (e.g. serial line) is wasted for debugging purposes. Even better, you can use fast Ethernet debugging with target systems without network capability. The host to BDI communication uses the standard GDB remote protocol.

An additional Telnet interface is available for special debug tasks (e.g. force a hardware reset, program flash memory).

The following figure shows how the BDI2000 interface is connected between the host and the target:



1.1 BDI2000

The BDI2000 is the main part of the bdiGDB system. This small box implements the interface between the JTAG pins of the target CPU and a 10Base-T ethernet connector. The firmware and the programable logic of the BDI2000 can be updated by the user with a simple Windows / Linux configuration program. The BDI2000 supports 1.8 – 5.0 Volts target systems (3.0 – 5.0 Volts target systems with Rev. A/B).

1.2 BDI Configuration

As an initial setup, the IP address of the BDI2000, the IP address of the host with the configuration file and the name of the configuration file is stored within the flash of the BDI2000.
Every time the BDI2000 is powered on, it reads the configuration file via TFTP.

Following an example of a typical configuration file:

```
; Configuration file for a MC68360 board
; --------------------------------------
; the initialistion list used to setup the target system
[INIT]
WR DFC        7           ;Set CPU space to access MBAR
WL 0x0003FF00 0x07000133  ;MBAR: dual port RAM / internal registers at 0x07000000
WR DFC        5           ;Select supervisor data space
WR SFC        5           ;Select supervisor data space
WB 0x07001009 0xFF        ;RSR: Clear reset status register
WB 0x07001022 0x04        ;SYPCR: disable watchdog, enable bus monitor (1k clocks)
WL 0x07001000 0x00004CFF  ;MCR: FRZ1, BCLR=3, SUPV, BCLRI=7, IARB=15
WW 0x07001010 0x0000E2F9  ;PLLCR: enable PLL, lock access, div128, multiplication = 761
DELAY         1000        ;Delay after changing PLL frequency
WW 0x07001014 0x8000      ;CDVCR: lock access
WB 0x0700100C 0x8F        ;CLKOCR: disable CLKO2, disable CLKO1, lock access
WW 0x07001016 0x0080      ;PEPAR: WE3-WE0, OE, CAS0...CAS3,CS7,AVEC
WL 0x07001040 0x0C900120  ;GMR: refresh 2banks, 2M, no CPU space, GAMX
WL 0x07001054 0x2FF80004  ;OR0: Boot Flash, 512k, 1wait, 8bit
WL 0x07001050 0x05000001  ;BR0: @05000000
WL 0x07001064 0x0FC00001  ;OR1: DRAM 4M, 3cycle access, 32bit
WL 0x07001060 0x00000001  ;BR1: @00000000

[TARGET]
CPUCLOCK      25000000     ;the CPU clock rate after processing the init list
BREAKMODE     SOFT         ;SOFT or HARD, HARD needs a processor with a SIM60
SIM60         0x07001000   ;the base address of the SIM60 module (for HW breakpoints)

[HOST]
IP          151.120.25.114
FILE        C:\cygnus\b19\demo\cpu32\aout
FORMAT      AOUT
LOAD        MANUAL      ;load code MANUAL or AUTO after reset

[FLASH]
CHIPTYPE    AM29F       ;Flash type (AM29F | AM29BX8 | AM29BX16 | I28BX8 | I28BX16)
CHIPSIZE    0x80000     ;The size of one flash chip in bytes (e.g. AM29F010 = 0x20000)
BUSWIDTH    8           ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE        C:\cygnus\b19\demo\cpu32\bootrom.hex  ;The file to program
ERASE       0x05040000  ;erase sector 4 of flash
ERASE       0x05050000  ;erase sector 5 of flash
ERASE       0x05060000  ;erase sector 6 of flash
ERASE       0x05070000  ;erase sector 7 of flash
```

Based on the information in the configuration file, the target is initialized and the program file is loaded via TFTP.

2 Installation

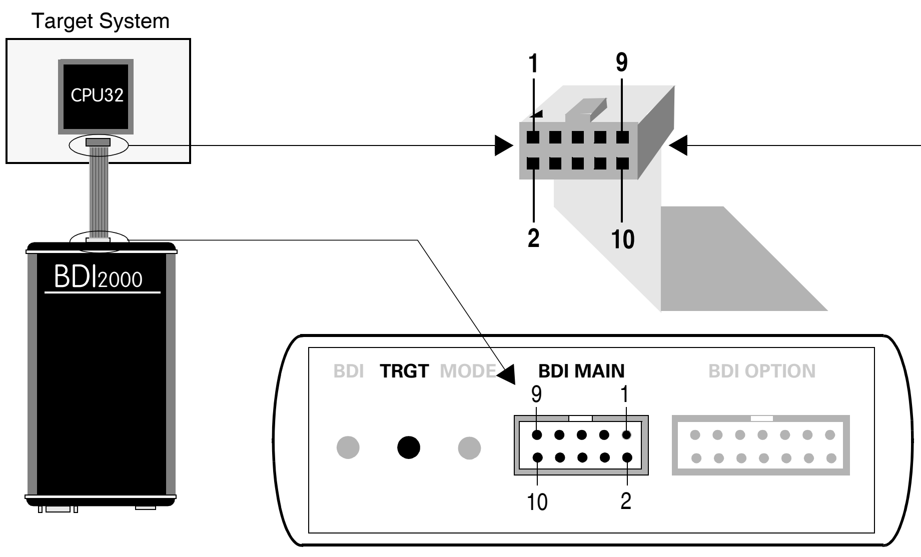
2.1 Connecting the BDI2000 to Target

The cable to the target system is a ten pin flat ribbon cable. In case where the target system has an appropriate connector, the cable can be directly connected. The pin assignment is in accordance with the Motorola specification.

⚠

In order to ensure reliable operation of the BDI (EMC, runtimes, etc.) the target cable length must not exceed 20 cm (8").

Rev. A ***«Rev. A» is the first BDI2000 version, produced until June 1999***

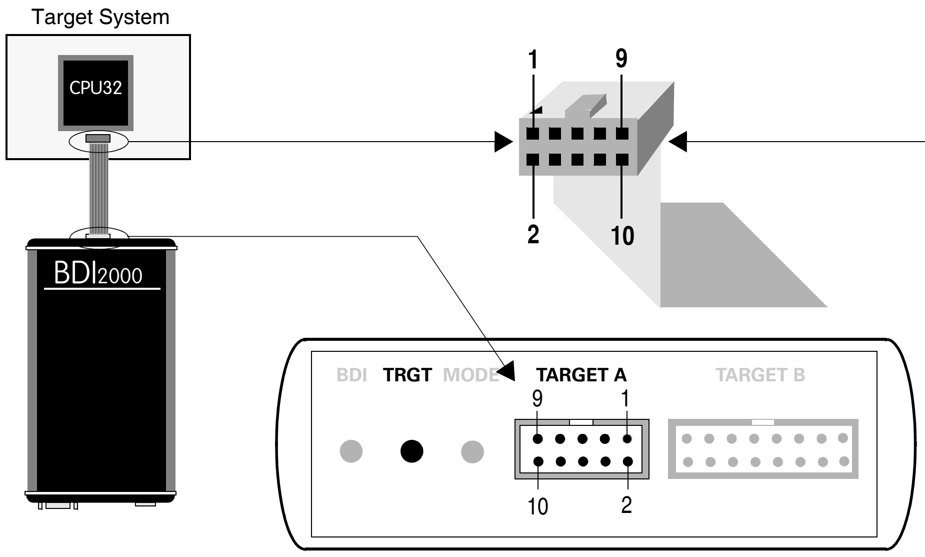


Target Connector

1 - NOT USED
2 - $\overline{\text{BERR}}$
3 - GROUND
4 - $\overline{\text{BKPT}}$ / DSCLK
5 - GROUND
6 - FREEZE
7 - $\overline{\text{RESET}}$
8 - DSDI
9 - Vcc Target
10 - $\overline{\text{IPIPE}}$ / DSO

The green LED «TRGT» marked light up when target is powered up

Rev. B/C



Target Connector

1 - NOT USED
2 - $\overline{\text{BERR}}$
3 - GROUND
4 - $\overline{\text{BKPT}}$ / DSCLK
5 - GROUND
6 - FREEZE
7 - $\overline{\text{RESET}}$
8 - DSDI
9 - Vcc Target
10 - $\overline{\text{IPIPE}}$ / DSO

The green LED «TRGT» marked light up when target is powered up

For BDI MAIN / TARGET A connector signals see table on next page.

BDI MAIN / TARGET A Connector Signals:

Pin	Name	Describtion
1	---	Not used.
2	$\overline{\text{BERR}}$	**BUS ERROR** Active-low input to the MCU. Signals an invalid bus operation attempt.
3+5	GROUND	**SYSTEM GROUND**
4	$\overline{\text{BKPT}}$ / DSCLK	**BREAKPOINT** For normal modes, active-low input to the MCU. Signals a hardware breakpoint. **DEVELOPMENT SERIAL CLOCK** For background debug mode, serial input clock signal to the MCU.
6	FREEZE	**FREEZE** Active-high output from the MCU. Indicates that the MCU has acknowledged a breakpoint and that it has entered background debug mode.
7	$\overline{\text{RESET}}$	**RESET** Active-low, open-drain, signal to start a system reset.
8	$\overline{\text{IFETCH}}$ / DSI	**INSTRUCTION FETCH** For normal modes, output signal from the MCU. Indicates instruction pipeline activity. **DATA SERIAL IN** For background debug mode, serial data input signal to the MCU.
9	Vcc Target	**1.8 – 5.0V:** This is the target reference voltage. It indicates that the target has power and it is also used to create the logic-level reference for the input comparators. It also controls the output logic levels to the target. It is normally fed from Vdd I/O on the target board. **3.0 – 5.0V with Rev. A/B :** This input to the BDI2000 is used to detect if the target is powered up. If there is a current limiting resistor between this pin and the target Vdd, it should be 100 Ohm or less.
10	$\overline{\text{IPIPE}}$ / DSO	**INSTRUCTION PIPE** For normal modes, output signal from the MCU. Indicates instruction pipeline activity. **DATA SERIAL OUT** For background debug mode, serial data output from the MCU.

All the pins except pin 1need to be connected to the target system for the debug operation.

2.1.1 Changing Target Processor Type

Before you can use the BDI2000 with an other target processor type (e.g. CPU32 <--> PPC), a new setup has to be done (see chapter 2.5). During this process the target cable must be disconnected from the target system. The BDI2000 needs to be supplied with 5 Volts via the BDI OPTION connector (Version A) or via the POWER connector (Version B). For more information see chapter 2.2.1 «External Power Supply».

⚠

To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU.

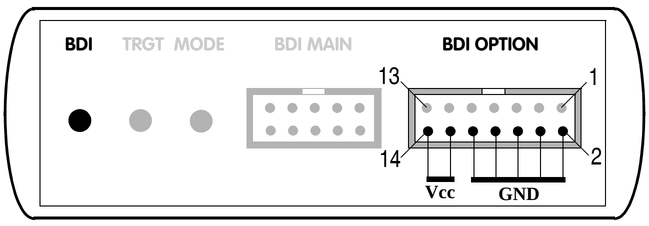
2.2 Connecting the BDI2000 to Power Supply

2.2.1 External Power Supply

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via the BDI OPTION connector (Rev. A) or via POWER connector (Rev. B/C). The available power supply from Abatron (option) or the enclosed power cable can be directly connected. In order to ensure reliable operation of the BDI2000, keep the power supply cable as short as possible.

For error-free operation, the power supply to the BDI2000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**

Rev. A

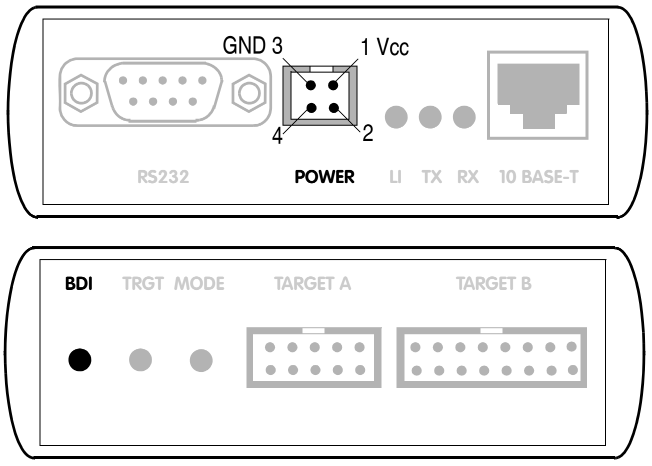


The green LED «BDI» marked light up when 5V power is connected to the BDI2000

BDI OPTION Connector

1 - NOT USED
2 - GROUND
3 - NOT USED
4 - GROUND
5 - NOT USED
6 - GROUND
7 - NOT USED
8 - GROUND
9 - NOT USED
10 - GROUND
11 - NOT USED
12 - Vcc (+5V)
13 - Vcc Target (+5V)
14 - Vcc (+5V)

Rev. B/C



The green LED «BDI» marked light up when 5V power is connected to the BDI2000

POWER Connector

1 - Vcc (+5V)
2 - VccTGT
3 - GROUND
4 - NOT USED

Please switch on the system in the following sequence:

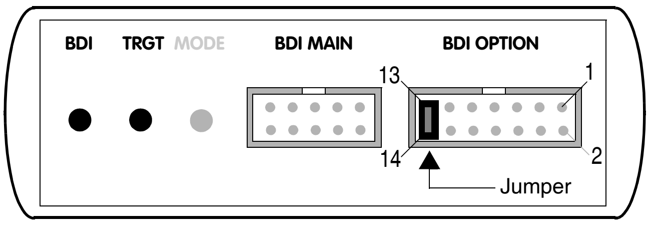
- 1 --> external power supply
- 2 --> target system

2.2.2 Power Supply from Target System

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via BDI MAIN target connector (Rev. A) or via TARGET A connector (Rev. B/C). This mode can only be used when the target system runs with 5V and the pin «Vcc Target» is able to deliver a current up to 1A@5V. For pin description and layout see chapter 2.1 «Connecting the BDI2000 to Target». Insert the enclosed Jumper as shown in figure below. **Please ensure that the jumper is inserted correctly**.

For error-free operation, the power supply to the BDI2000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**

Rev. A

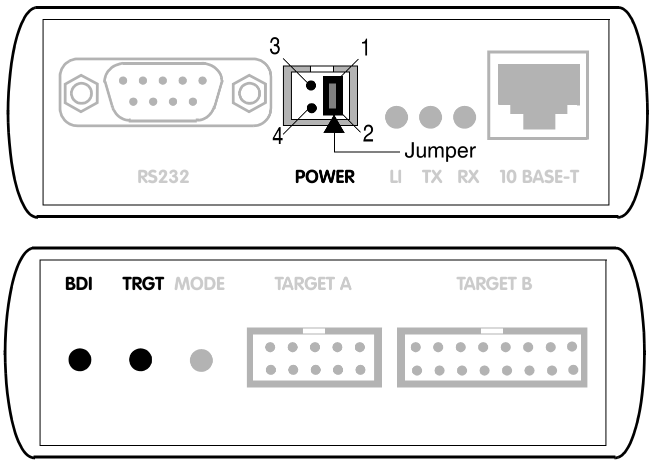


The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

BDI OPTION Connector

1 - NOT USED
2 - GROUND
3 - NOT USED
4 - GROUND
5 - NOT USED
6 - GROUND
7 - NOT USED
8 - GROUND
9 - NOT USED
10 - GROUND
11 - NOT USED
12 - Vcc (+5V)
13 - Vcc Target (+5V)
14 - Vcc BDI2000 (+5V)

Rev. B/C



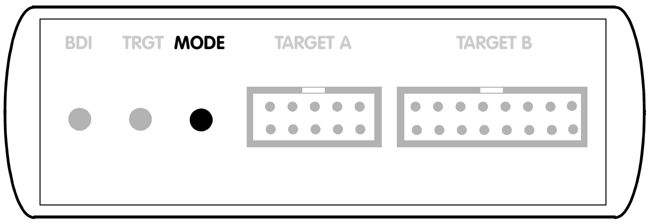
The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

POWER Connector

1 - Vcc BDI2000 (+5V)
2 - Vcc Target (+5V)
3 - GROUND
4 - NOT USED

2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



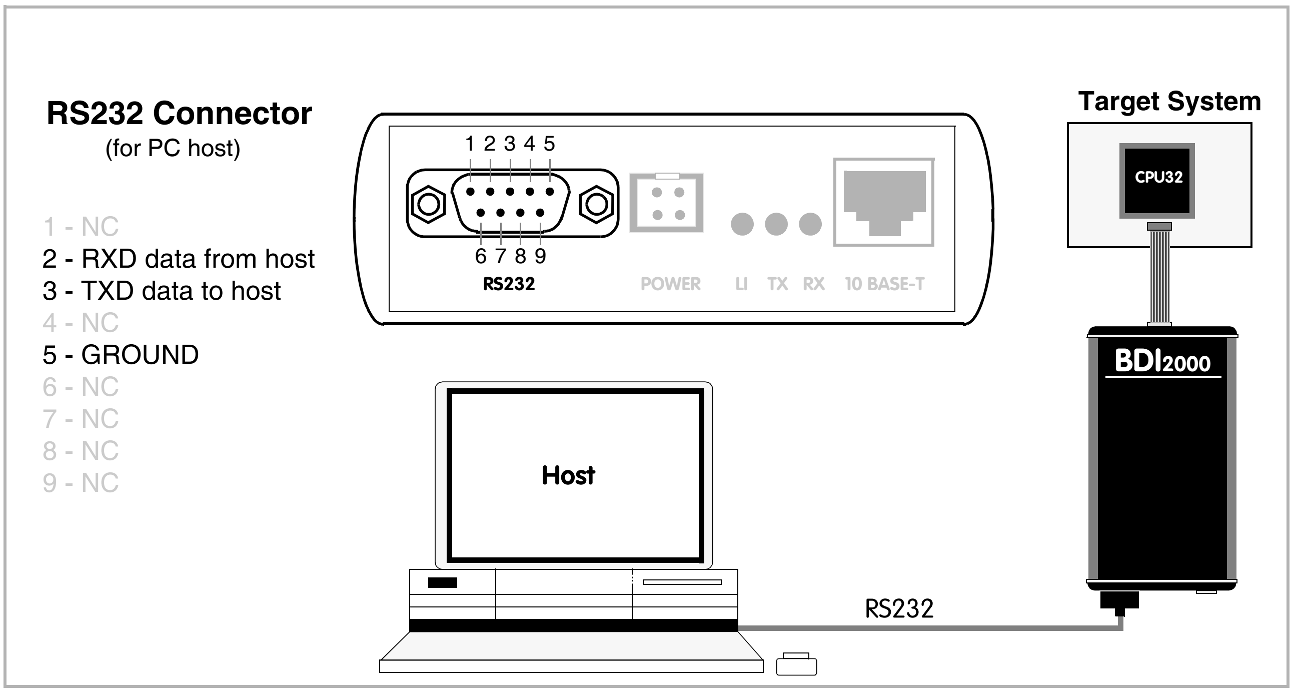
MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The power supply for the BDI2000 is < 4.75VDC.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

2.4 Connecting the BDI2000 to Host

2.4.1 Serial line communication

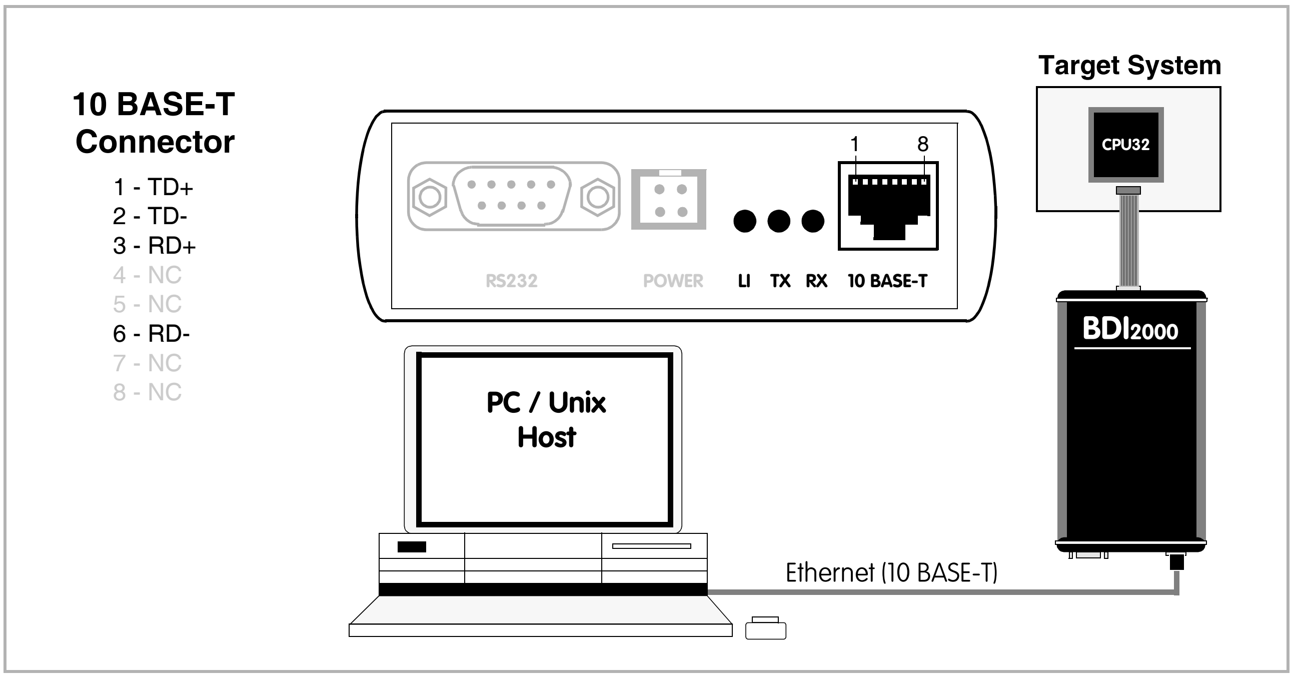
Serial line communication is only used for the initial configuration of the bdiGDB system.

The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable (included) between BDI and Host is a serial cable. There is the same connector pinout for the BDI and for the Host side (Refer to Figure below).



2.4.2 Ethernet communication

The BDI2000 has a built-in 10 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshilded Twisted Pair) cable to the BD2000. For thin Ethernet coaxial networks you can connect a commercially available media converter (BNC-->10 BASE-T) between your network and the BDI2000. Contact your network administrator if you have questions about the network.



The following explains the meanings of the built-in LED lights:

LED	Name	Description
LI	Link	When this LED light is ON, data link is successful between the UTP port of the BDI2000 and the hub to which it is connected.
TX	Transmit	When this LED light BLINKS, data is being transmitted through the UTP port of the BDI2000
RX	Receive	When this LED light BLINKS, data is being received through the UTP port of the BDI2000

2.5 Installation of the Configuration Software

On the enclosed CD you will find the BDI configuration software and the firmware required for the BDI2000. For Windows NT users there is also a TFTP server included.

The following files are on the CD.

b20c32gd.exe	Configuration program (16bit Windows application)
b20c32gd.hlp	Windows help file for the configuration program
b20c32gd.xxx	Firmware for the BDI2000
c32jed20.xxx	JEDEC file for the BDI2000 (Rev. A/B) logic device when working with a CPU32 target
c32jed21.xxx	JEDEC file for the BDI2000 (Rev. C) logic device when working with a CPU32 target
tftpsrv.exe	TFTP server for WindowsNT/ Windows95 (WIN32 console application)
cpu360.cfg	A sample configuration file for a MC68360
*.def	Register definition files
bdisetup.zip	ZIP Archive with the Setup Tool sources for Linux / UNIX hosts.

Overview of an installation / configuration process:

- Create a new directory on your hard disk
- Copy the entire contents of the enclosed CD into this directory
- Linux only: extract the setup tool sources and build the setup tool
- Use the setup tool to load/update the BDI firmware/logic
  Note: A new BDI has no firmware/logic loaded.
- Use the setup tool to transmit the initial configuration parameters
  - IP address of the BDI.
  - IP address of the host with the configuration file.
  - Name of the configuration file. This file is accessed via TFTP.
  - Optional network parameters (subnet mask, default gateway).

Activating BOOTP:
The BDI can get the network configuration and the name of the configuration file also via BOOTP. For this simple enter 0.0.0.0 as the BDI's IP address (see following chapters). If present, the subnet mask and the default gateway (router) is taken from the BOOTP vendor-specific field as defined in RFC 1533.

With the Linux setup tool, simply use the default parameters for the -c option:
```
[root@LINUX_1 bdisetup]# ./bdisetup -c -p/dev/ttyS0 -b57
```

The MAC address is derived from the serial number as follows:
MAC: 00-0C-01-xx-xx-xx , repace the xx-xx-xx with the 6 left digits of the serial number
Example: SN# 93123457 ==>> 00-0C-01-93-12-34

2.5.1 Configuration with a Linux / Unix host

The firmware / logic update and the initial configuration of the BDI2000 is done with a command line utility. In the ZIP Archive bdisetup.zip are all sources to build this utility. More information about this utility can be found at the top in the bdisetup.c source file. There is also a make file included. Starting the tool without any parameter displays information about the syntax and parameters.

To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU (see Chapter 2.1.1).

Following the steps to bring-up a new BDI2000:

1. Build the setup tool:
The setup tool is delivered only as source files. This allows to build the tool on any Linux / Unix host. To build the tool, simply start the make utility.

```
[root@LINUX_1 bdisetup]# make
cc -O2   -c -o bdisetup.o bdisetup.c
cc -O2   -c -o bdicnf.o bdicnf.c
cc -O2   -c -o bdidll.o bdidll.c
cc -s bdisetup.o bdicnf.o bdidll.o -o bdisetup
```

2. Check the serial connection to the BDI:
With "bdisetup -v" you may check the serial connection to the BDI. The BDI will respond with information about the current loaded firmware and network configuration.
Note: Login as root, otherwise you probably have no access to the serial port.

```
[root@LINUX_1 bdisetup]# ./bdisetup -v -p/dev/ttyS0 -b57
BDI Type : BDI2000 Rev.C (SN: 92152150)
Loader   : V1.05
Firmware : unknown
Logic    : unknown
MAC      : ff-ff-ff-ff-ff-ff
IP Addr  : 255.255.255.255
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 255.255.255.255
Config   : ??????????????????
```

3. Load/Update the BDI firmware/logic:
With "bdisetup -u" the firmware is loaded and the CPLD within the BDI2000 is programmed. This configures the BDI for the target you are using. Based on the parameters -a and -t, the tool selects the correct firmware / logic files. If the firmware / logic files are in the same directory as the setup tool, there is no need to enter a -d parameter.

```
[root@LINUX_1 bdisetup]# ./bdisetup -u -p/dev/ttyS0 -b57 -aGDB -tCPU32
Connecting to BDI loader
Erasing CPLD
Programming firmware with ./b20c32gd.101
Programming CPLD with ./c32jed21.102
```

4. Transmit the initial configuration parameters:
With "bdisetup -c" the configuration parameters are written to the flash memory within the BDI.
The following parameters are used to configure the BDI:

BDI IP Address	The IP address for the BDI2000. Ask your network administrator for assigning an IP address to this BDI2000. Every BDI2000 in your network needs a different IP address.
Subnet Mask	The subnet mask of the network where the BDI is connected to. A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask. If the BDI and the host are in the same subnet, it is not necessary to enter a subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI2000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This file is read via TFTP. Keep in mind that TFTP has it's own root directory (usual /tftpboot). You can simply copy the configuration file to this directory and the use the file name without any path. For more information about TFTP use "man tftpd".

```
[root@LINUX_1 bdisetup]# ./bdisetup -c -p/dev/ttyS0 -b57 \
> -i151.120.25.101 \
> -h151.120.25.118 \
> -fcpu360.cnf
Connecting to BDI loader
Writing network configuration
Writing init list and mode
Configuration passed
```

5. Check configuration and exit loader mode:
The BDI is in loader mode when there is no valid firmware loaded or you connect to it with the setup tool. While in loader mode, the Mode LED is flashing. The BDI will not respond to network requests while in loader mode. To exit loader mode, the "bdisetup -v -s" can be used. You may also power-off the BDI, wait some time (1min.) and power-on it again to exit loader mode.

```
[root@LINUX_1 bdisetup]# ./bdisetup -v -p/dev/ttyS0 -b57 -s
BDI Type : BDI2000 Rev.C (SN: 92152150)
Loader   : V1.05
Firmware : V1.01 bdiGDB for CPU32
Logic    : V1.02 CPU32/CPU16
MAC      : 00-0c-01-92-15-21
IP Addr  : 151.120.25.101
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 151.120.25.118
Config   : cpu360.cnf
```

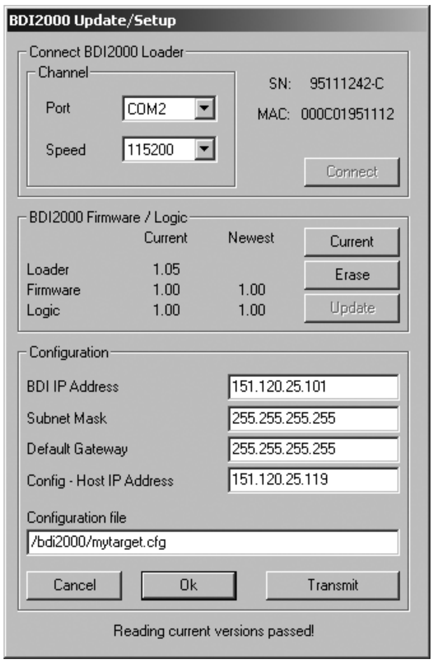
The Mode LED should go off, and you can try to connect to the BDI via Telnet.

```
[root@LINUX_1 bdisetup]# telnet 151.120.25.101
```

2.5.2 Configuration with a Windows host

First make sure that the BDI is properly connected (see Chapter 2.1 to 2.4).

To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU (see Chapter 2.1.1).


BDI2000 Update/Setup

Connect BDI2000 Loader

Channel

Port: COM2

Speed: 115200

SN: 95111242-C

MAC: 000C01951112

BDI2000 Firmware / Logic

	Current	Newest
Loader	1.05	
Firmware	1.00	1.00
Logic	1.00	1.00

Configuration

BDI IP Address: 151.120.25.101

Subnet Mask: 255.255.255.255

Default Gateway: 255.255.255.255

Config - Host IP Address: 151.120.25.119

Configuration file: /bdi2000/mytarget.cfg

Reading current versions passed!

dialog box «BDI2000 Update/Setup»

Before you can use the BDI2000 together with the GNU debugger, you must store the initial configuration parameters in the BDI2000 flash memory. The following options allow you to do this:

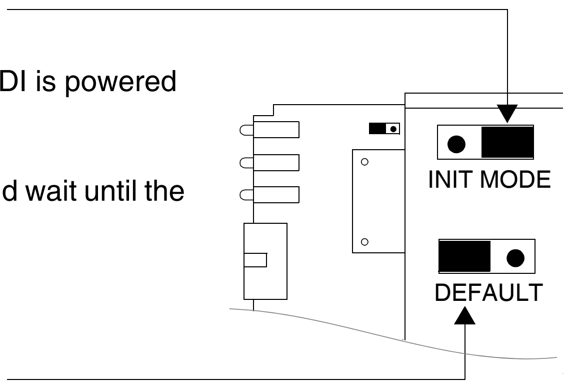
Port	Select the communication port where the BDI2000 is connected during this setup session.
Speed	Select the baudrate used to communicate with the BDI2000 loader during this setup session.
Connect	Click on this button to establish a connection with the BDI2000 loader. Once connected, the BDI2000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI2000 software and logic versions. The current loader, firmware and logic version will be displayed.
Update	This button is only active if there is a newer firmware or logic version present in the execution directory of the bdiGDB setup software. Press this button to write the new firmware and/or logic into the BDI2000 flash memory / programmable logic.

BDI IP Address	Enter the IP address for the BDI2000. Use the following format: xxx.xxx.xxx.xxx e.g.151.120.25.101 Ask your network administrator for assigning an IP address to this BDI2000. Every BDI2000 in your network needs a different IP address.
Subnet Mask	Enter the subnet mask of the network where the BDI is connected to. Use the following format: xxx.xxx.xxx.xxxe.g.255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value..
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI2000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. e.g. D:\gnu\config\bdi\ads8260bdi.cnf For information about the syntax of the configuration file see the bdiGDB User manual. This name is transmitted to the TFTP server when reading the configuration file.
Transmit	Click on this button to store the configuration in the BDI2000 flash memory.

2.5.3 Recover procedure

In rare instances you may not be able to load the firmware in spite of a correctly connected BDI (error of the previous firmware in the flash memory). **Before carrying out the following procedure, check the possibilities in Appendix «Troubleshooting»**. In case you do not have any success with the tips there, do the following:

- Switch OFF the power supply for the BDI and open the unit as described in Appendix «Maintenance»
- Place the jumper in the **«INIT MODE»** position
- Connect the power cable or target cable if the BDI is powered from target system
- Switch ON the power supply for the BDI again and wait until the LED «MODE» blinks fast
- Turn the power supply OFF again
- Return the jumper to the **«DEFAULT»** position
- Reassemble the unit as described in Appendix «Maintenance»



2.6 Testing the BDI2000 to host connection

After the initial setup is done, you can test the communication between the host and the BDI2000. There is no need for a target configuration file and no TFTP server is needed on the host.

- If not already done, connect the bdiGDB system to the network.
- Power-up the BDI2000.
- Start a Telnet client on the host and connect to the BDI2000 (the IP address you entered during initial configuration).
- If everything is okay, a sign on message like «BDI Debugger for CPU32» should be displayed in the Telnet window.

2.7 TFTP server for Windows

The bdiGDB system uses TFTP to access the configuration file and to load the application program. Because there is no TFTP server bundled with Windows NT, Abatron provides a TFTP server application **tftpsrv.exe**. This WIN32 console application runs as normal user application (not as a system service).

Command line syntax: tftpsrv [p] [w] [dRootDirectory]

Without any parameter, the server starts in read-only mode. This means, only read access request from the client are granted. This is the normal working mode. The bdiGDB system needs only read access to the configuration and program files.

The parameter [p] enables protocol output to the console window. Try it.
The parameter [w] enables write accesses to the host file system.
The parameter [d] allows to define a root directory.

`tftpsrv p`	Starts the TFTP server and enables protocol output
`tftpsrv p w`	Starts the TFTP server, enables protocol output and write accesses are allowed.
`tftpsrv dC:\tftp\`	Starts the TFTP server and allows only access to files in C:\tftp and its subdirectories. As file name, use relative names. For example "bdi\mpc750.cfg" accesses "C:\tftp\bdi\mpc750.cfg"

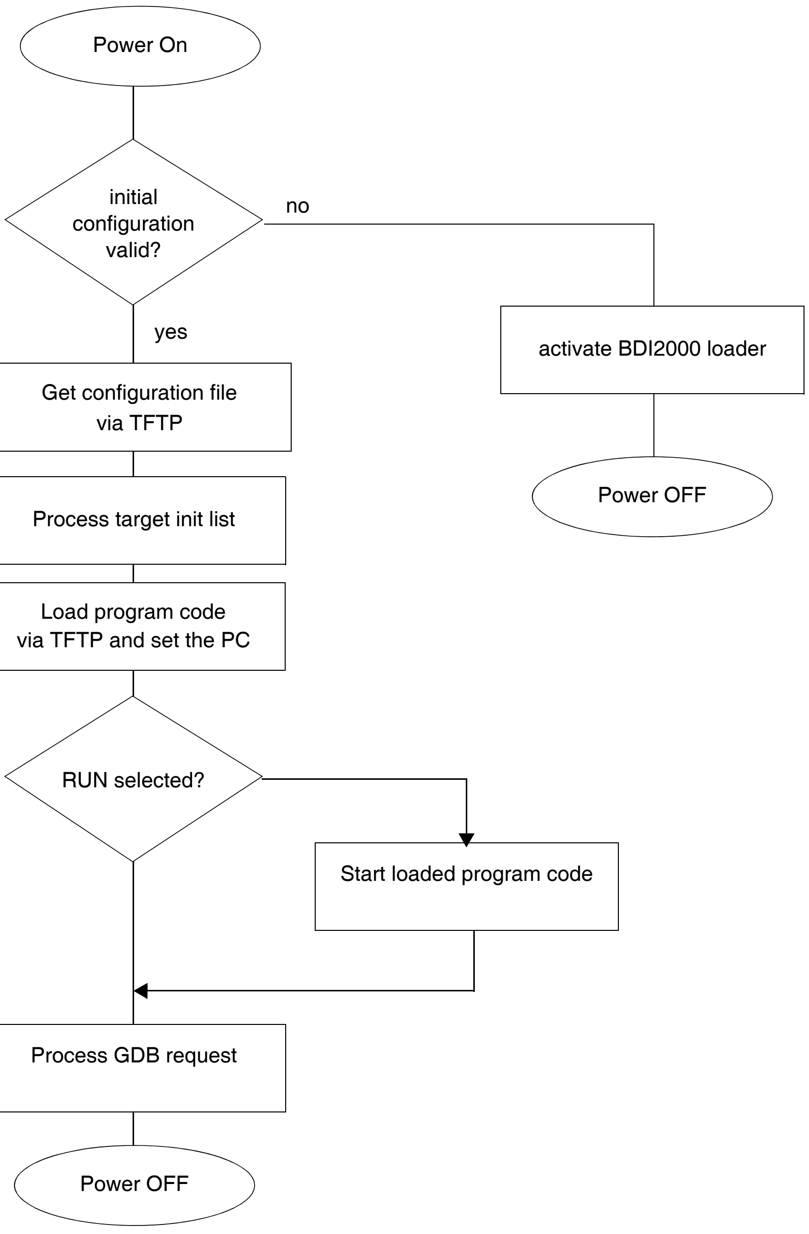
You may enter the TFTP server into the Startup group so the server is started every time you logon.

3 Using bdiGDB

3.1 Principle of operation

The firmware within the BDI handles the GDB request and accesses the target memory or registers via the BDM interface. There is no need for any debug software on the target system. After loading the code via TFTP debugging can begin at the very first assembler statement.

Whenever the bdiGDB system is started (target is powered on) the following sequence starts:


Power On
initial configuration valid?
no
yes
activate BDI2000 loader
Power OFF
Get configuration file via TFTP
Process target init list
Load program code via TFTP and set the PC
RUN selected?
Start loaded program code
Process GDB request
Power OFF

Breakpoints:
Normally breakpoints are implemented by replacing application code with the BGND instruction. All the time the application is suspended (i.e. caused by a breakpoint) the target processor remains freezed. In the special case, when a SIM60 (e.g. MC68360) is present, the internal breakpoint logic is also supported. This allows setting a breakpoint (watchpoint) even when the code is in read only memory (check MC68360 device errata for restriction concerning the SIM60 breakpoint logic).

SIM60 Hardware Breakpoints:
The SIM60 (MC68360) breakpoint logic is also supported. To enable the use of this special breakpoint hardware, set the following configuration parameters.

```
SIM60        0x07001000   ;The base address of the SIM60 module
```

Target Exceptions:
If the vector base register (VBR) is set with an entry in the initialization list, the BDI will catch all unhandled exceptions (e.g. Zero Division). This is only possible if the vector table is writable. At vector 0 the BDI writes a BGND, RTE instruction sequence and lets all other vectors point to this short exception handler.

```
WR VBR        0x00000000  ;set vector base and enable exception catching
```

3.2 Configuration File

The configuration file is automatically read by the BDI2000 after every power on.
The syntax of this file is as follows:

```
; comment
[part name]
identifier parameter1 parameter2 ..... parameterN  ; comment
identifier parameter1 parameter2 ..... parameterN
.....
[part name]
identifier parameter1 parameter2 ..... parameterN
identifier parameter1 parameter2 ..... parameterN
.....
             etc.
```

Numeric parameters can be entered as decimal (e.g. 700) or as hexadecimal (0x80000).

3.2.1 Part [INIT]

The part [INIT] defines a list of commands which should be executed every time the target comes out of reset. The commands are used to get the target ready for loading the program file. The SIM registers (chip select, clock, ...) are usually initialized with this command list.

WR register value	Write value to the selected register. register — D0...D7, A0...A7, PC, SR, USP, SSP, SFC, DFC, VBR value — the value to write into the register Example: WR SFC 5 ;set supervisor data space
WB address value	Write a byte (8bit) to the selected memory place. address — the memory address value — the value to write to the target memory Example: WB 0xFFFFFA21 0x04 ; SYPCR: watchdog disable ...
WW address value	Write a word (16bit) to the selected memory place. address — the memory address value — the value to write to the target memory Example: WW 0xFFFFFA04 0x7F00 ;SYNCR: w=0, x=1 -> 16.7 MHz
WL address value	Write a long (32bit) to the selected memory place. address — the memory address value — the value to write to the target memory Example: WL 0xFFFFFA48 0x00077870 ; CSBOOT 1MB, 1wait, ...
DELAY value	Delay for the selected time. A delay may be necessary to let the clock PLL lock again after a new clock rate is selected. value — the delay time in milliseconds (1...30000) Example: DELAY 500 ; delay for 0.5 seconds

At the beginning of every command list, the DFC and the SFC should be set to supervisor data space. This because all memory accesses via BDM uses the memory space selected in DFC and SFC.

3.2.2 Part [TARGET]

The part [TARGET] defines some target specific values and the BDI operating mode.

BDIMODE mode param	This parameter selects the BDI debugging mode. The following modes are supported:	
	LOADONLY	Loads and starts the program file. No debugging via BDM.
	AGENT	Normal debugging mode. There is no need for any debug software on the target. This mode accepts a second parameter. If RUN is entered as a second parameter, the loaded application will be started immediately, otherwise only the PC is set and BDI waits for GDB requests.
	Example:	BDIMODE AGENT RUN
CPUCLOCK value	The BDI needs to know how fast the target CPU runs after processing the init list. The BDM communication speed is selected based on this value. If this value defines a clock rate that is higher than the real clock, BDM communication may fail. When defining a clock rate slower than possible, BDM communication still works but not as fast as possible.	
	value	the CPU clock in hertz
	Example:	CPUCLOCK 16777000 ;CPU clock is 16.7MHz
SIM60 value	If a SIM60 module is present, the value defines where the SIM60 registers are located in memory. Do not set this parameter if there is no SIM60 available.	
	value	the base address of the SIM60 register file.
	Example:	SIM60 0x07001000 ; MC68360 SIM register address
BREAKMODE mode	This parameter defines how breakpoints are implemented and is only used if a SIM60 module is present.	
	SOFT	This is the normal mode. Breakpoints are implemented by replacing code with a BGND instruction.
	HARD	In this mode, the SIM60 breakpoint hardware is used. Only one breakpoint at a time is supported. Keep in mind, that in this mode, the CPU stops after executing the opcode at the breakpoint address.
	Example:	BREAKMODE HARD ; enable use of break hardware

3.2.3 Part [HOST]

The part [HOST] defines some host specific values.

IP ipaddress — The IP address of the host where the program file is located.

ipaddress	the IP address in the form xxx.xxx.xxx.xxx
Example:	IP 151.120.25.100

FILE filename — The file name of the program file. This name is used to access the program file via TFTP.

filename	the filename including the full path
Example:	FILE C:\gnu\target\cpu360\a.out

FORMAT format [offset] — The format of the image file and an optional load address offset. If the image is already stored in ROM on the target, select ROM as the format. The optional parameter "offset" is added to any load address read from the image file.

format	SREC, BIN, AOUT, ELF, COFF or ROM
Example:	FORMAT ELF FORMAT ELF 0x10000

DEBUGPORT port [RECONNECT] — The TCP port GDB uses to access the target. If the RECONNECT parameter is present, an open TCP/IP connection (Telnet/GDB) will be closed if there is a connect request from the same host (same IP address).

port	the TCP port number (default = 2001)
Example:	DEBUGPORT 2001

LOAD mode — In Agent mode, this parameters defines if the code is loaded automatically after every reset.

mode	AUTO, MANUAL
Example:	LOAD MANUAL

START address — The address where to start the loaded application. If this value is not defined and the application is not in ROM, the lowest loaded memory address will be used. If this value is not defined and the application is already in ROM, the PC will not be set before starting the application. This means, the start vector loaded at reset time will be used.

address	the address where to start the application
Example:	START 0x1000

PROMPT string — This entry defines a new Telnet prompt. The current prompt can also be changed via the Telnet interface.

Example:	PROMPT 331>

DUMP filename — The default file name used for the Telnet DUMP command.

filename	the filename including the full path
Example:	DUMP dump.bin

TELNET mode — By default the BDI sends echoes for the received characters and supports command history and line editing. If it should not send echoes and let the Telnet client in "line mode", add this entry to the configuration file.

mode	ECHO (default), NOECHO or LINE
Example:	TELNET NOECHO ; use old line mode

3.2.4 Part [FLASH]

The Telnet interface supports programming and erasing of flash memories. The bdiGDB system has to know which type of flash is used, how the chip(s) are connected to the CPU and which sectors to erase in case the ERASE command is entered without any parameter.

CHIPTYPE type	This parameter defines the type of flash used. It is used to select the correct programming algorithm. format AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16 Example: CHIPTYPE AM29F
CHIPSIZE size	The size of **one** flash chip in bytes (e.g. AM29F010 = 0x20000). This value is used to calculate the starting address of the current flash memory bank. size the size of one flash chip in bytes Example: CHIPSIZE 0x80000
BUSWIDTH width	Enter the width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter CHIPTYPE carries the information about the number of data lines connected to one flash chip. For example, enter 16 if you are using two AM29F010 to build a 16bit flash memory bank. with the width of the flash memory bus in bits (8 | 16 | 32) Example: BUSWIDTH 16
FILE filename	The name of the file to program into the flash. This name is used to access the file via TFTP. This name may be overridden interactively at the Telnet interface. filename the filename including the full path Example: FILE C:\gnu\target\cpu360\bootrom.hex
FORMAT format [offset]	The format of the file and an optional address offset. The optional parameter "offset" is added to any load address read from the program file. format SREC, BIN, AOUT, ELF or COFF Example: FORMAT SREC FORMAT ELF 0x10000
WORKSPACE address	If a workspace is defined, the BDI uses a faster programming algorithm that runs out of RAM on the target system. Otherwise, the algorithm is processed within the BDI. The workspace is used for a 1kByte data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose. address the address of the RAM area Example: WORKSPACE 0x00000000

ERASE address [mode]	The flash memory may be individually erased via the Telnet interface. In order to make erasing of multiple flash sectors easier, you can enter an erase list. All entries in the erase list will be processed if you enter ERASE at the Telnet prompt without any parameter.

address	Address of the flash sector, block or chip to erase
mode	BLOCK, CHIP Without this optional parameter, the BDI executes a sector erase. If supported by the chip, you can also specify a block or chip erase.
Example:	ERASE 0x05040000 ;erase sector 4 of flash
	ERASE 0x05060000 ;erase sector 6 of flash
	ERASE 0x05000000 CHIP ;erase whole chip(s)

Supported Flash Memories:
There are currently 3 standard flash algorithm supported. The AMD, Intel and Atmel AT49 algorithm. Almost all currently available flash memories can be programmed with one of this algorithm. The flash type selects the appropriate algorithm and gives additional information about the used flash.

For 8bit only flash, select:	AM29F, I28BX8 or AT49
For 8/16 bit flash in 8bit mode, select:	AM29BX8, I28BX8 or AT49X8
For 8/16 bit flash in 16bit mode, select:	AM29BX16, I28BX16 or AT49X16
For 16bit only flash, select:	AM29BX16, I28BX16 or AT49X16

The AMD and AT49 algorithm are almost the same. The only difference is, that the AT49 algorithm does not check for the AMD status bit 5 (Exceeded Timing Limits).
Only the AMD and AT49 algorithm support chip erase. Block erase is only supported with the AT49 algorithm. If the algorithm does not support the selected mode, sector erase is performed. If the chip does not support the selected mode, erasing will fail. The erase command sequence is different only in the 6th write cycle. Depending on the selected mode, the following data is written in this cycle (see also flash data sheets): 0x10 for chip erase, 0x30 for sector erase, 0x50 for block erase.

The following table shows some examples:

Flash	x 8	x 16	Chipsize
Am29F010	AM29F	-	0x020000
Am29F800B	AM29BX8	AM29BX16	0x100000
Am29DL323C	AM29BX8	AM29BX16	0x400000
Intel 28F032B3	I28BX8	-	0x400000
Intel 28F640J3A	I28BX8	I28BX16	0x800000
Intel 28F320C3	-	I28BX16	0x400000
AT49BV040	AT49	-	0x080000
AT49BV1614	AT49X8	AT49X16	0x200000
SST39VF160	-	AT49X16	0x200000

Note:
Some Intel flash chips (e.g. 28F800C3, 28F160C3, 28F320C3) power-up with all blocks in locked state. In order to erase/program those flash chips, use the init list to unlock the appropriate blocks.

```
WM16   0xFFF00000   0x0060       unlock block 0
WM16   0xFFF00000   0x00D0
WM16   0xFFF10000   0x0060       unlock block 1
WM16   0xFFF10000   0x00D0
       ....
WM16   0xFFF00000   0xFFFF       select read mode
```

3.2.5 Part [REGS]

In order to make it easier to access target registers via the Telnet interface, the BDI can read in a register definition file. In this file, the user defines a name for the register and how the BDI should access it (e.g. as memory mapped, memory mapped with offset, ...). The name of the register definition file and information for different registers type has to be defined in the configuration file.
The register name, type, address/offset/number and size are defined in a separate register definition file. This way, you can create one register definition file for the MC68360 that can be used for all possible positions of the internal memory map. You only have to change one entry in the configuration file.

An entry in the register definition file has the following syntax:

```
name   type   addr   size
```

name	The name of the register (max. 15 characters)
type	The register type MM Absolute direct memory mapped register DMM1...DMM4 Relative direct memory mapped register IMM1...IMM4 Indirect memory mapped register
addr	The address, offset or number of the register
size	The size (8, 16, 32) of the register

The following entries are supported in the [REGS] part of the configuration file:

FILE filename	The name of the register definition file. This name is used to access the file via TFTP. The file is loaded once during BDI startup. filename the filename including the full path Example: FILE C:\bdi\regs\cpu360.def
DMMn base	This defines the base address of direct memory mapped registers. This base address is added to the individual offset of the register. base the base address Example: DMM1 0x07001000
IMMn addr data	This defines the addresses of the memory mapped address and data registers of indirect memory mapped registers. The address of a IMMn register is first written to "addr" and then the register value is access using "data" as address. addr the address of the Address register data the address of the Data register Example: IMM1 0x07001040 0x07001044

Example for a register definition (MC68360):

Entry in the configuration file:

```
[REGS]
DMM1   0x07001000                       ;Internal Memory Map Base Address
FILE   E:\bdi\cpu360\reg360.def         ;The register definition file
```

The register definition file:

```
;Register definition for MC68360
;===============================
;
; name: user defined name of the register
; type: the type of the register
;      MM     memory mapped register
;      DMMx   direct memory mapped register with offset
;             x = 1..4
;             the base is defined in the configuration file
;             e.g. DMM1 0x07001000
; addr:the number, adddress or offset of the register
; size the size of the register (8,16 or 32)
;
;name         type   addr         size
;-------------------------------------------
;
; DMM1 must be set to the internal register base address
;
mcr           DMM1   0x0000       32
avr           DMM1   0x0008       8
rsr           DMM1   0x0009       8
```

Now the defined registers can be accessed by name via the Telnet interface:

```
BDI> rd mcr
BDI>rm avr 0xFF
```

Note:
The CPU registers D0...D7, A0...A7, PC, SR, USP, SSP, SFC, DFC, VBR are already predefined.

3.3 Debugging with GDB

Because the target agent runs within BDI, no debug support has to be linked to your application. There is also no need for any BDI specific changes in the application sources. Your application must be fully linked because no dynamic loading is supported.

3.3.1 Target setup

Target initialization may be done at two places. First with the BDI configuration file, second within the application. The setup in the configuration file must at least enable access to the target memory where the application will be loaded. Disable the watchdog and setting the CPU clock rate should also be done with the BDI configuration file. Application specific initializations like setting the timer rate are best located in the application startup sequence.

Following two examples for possible target initializations.

```
; this initialisation list was used for a Motorola evaluation board
; with a M68F333 and 512kByte of SRAM
[INIT]
WR DFC        5          ;Select supervisor data space
WR SFC        5          ;Select supervisor data space
WB 0xFFFFFA21 0x04       ;SYPCR: watch dog disabled, BM enabled 64 clk
WW 0xFFFFFA04 0x7F00     ;SYNCR w=0,x=1 -> 16.7MHz
DELAY         700        ;Delay for 0.7 seconds
WW 0xFFFFFB04 0x2000     ;RAMBAR: internal RAM at 200000
WW 0xFFFFFB00 0x0000     ;RAMMCR: enable internal RAM
WL 0xFFFFFA48 0x00057870 ;CSBOOT: 256k at 0x000000, 1wait, r/w, both
WL 0xFFFFFA4C 0x04055070 ;CS0: RAM write(MSB), 256k at 0x40000
WL 0xFFFFFA58 0x04053070 ;CS3: RAM write(LSB), 256k at 0x40000
WL 0xFFFFFA60 0x04056870 ;CS5: RAM read (word access), 256k at 0x40000
```

```
; this initialisation list was used for a MC68360 board
[INIT]
WR DFC        7          ;Select CPU space to access MBAR
WL 0x0003FF00 0xFFFFE001 ;MBAR: dual port RAM / internal register at 0xFFFFE000
WR DFC        5          ;Select supervisor data space
WR SFC        5          ;Select supervisor data space
WB 0xFFFFF022 0x04       ;SYPCR:  watch dog disabled, BM enabled 1 kClk
WB 0xFFFFF00C 0x0F       ;CLKOCR: disable clock output
WW 0xFFFFF016 0x0080     ;PEPAR:  WE3-WE0
WL 0xFFFFF040 0x00000100 ;GMR: no CPU space
WL 0xFFFFF054 0x4FF00004 ;OR0: Boot EPROM 1M, 3wait, 8bit
WL 0xFFFFF050 0x00000003 ;BR0: at 0x00000000, write protect
WL 0xFFFFF064 0x5FE00000 ;OR1: SRAM 2M, 2wait, 32bit
WL 0xFFFFF060 0x00400001 ;BR1: at 0x00400000
WL 0xFFFFF084 0x3FE00000 ;OR3: FLASH 2M, 2wait, 32bit
WL 0xFFFFF080 0x00600001 ;BR3: at 0x00600000
```

Setup a board with a CPU32 is not an easy job. For information about your hardware consult the appropriate manuals from Motorola or from your board supplier.

3.3.2 Connecting to the target

As soon as the target comes out of reset, BDI initializes it and loads your application code. If RUN is selected, the application is immediately started, otherwise only the target PC is set. BDI now waits for GDB request from the debugger running on the host.

After starting the debugger, it must be connected to the remote target. This can be done with the following command at the GDB prompt:

```
(gdb)target remote bdi2000:2001
```

bdi2000	This stands for an IP address. The HOST file must have an appropriate entry. You may also use an IP address in the form xxx.xxx.xxx.xxx
2001	This is the TCP port used to communicate with the BDI

If not already suspended, this stops the execution of application code and the target CPU changes to background debug mode.

Remember, every time the application is suspended, the target CPU is freezed. During this time no hardware interrupts will be processed.

Note: For convenience, the GDB detach command triggers a target reset sequence in the BDI.

```
(gdb)...
(gdb)detach
... Wait until BDI has resetet the target and reloaded the image
(gdb)target remote bdi2000:2001
```

3.3.3 GDB monitor command

The BDI supports the GDB V5.x "monitor" command. Telnet commands are executed and the Telnet output is returned to GDB.

```
(gdb) mon rd
D0 : 00000030 0000000e 00000001 00000001
D4 : 00000001 00000080 00000001 00000001
A0 : 0000141d 00000001 00000001 00000001
A4 : 00000001 00000001 00000fe8 00001000
PC : 05001000     SR : 00002700
USP: 1204242c     SSP: 00001000
SFC: 00000005     DFC: 00000005
VBR: 00000000
(gdb)
```

3.3.4 Application Output to the Telnet and GDB console

It is possible to send text to the Telnet and GDB console from within the application code. For this the application code has to setup some registers and then execute a BGND instruction. Below an example how such a function can be implemented.

```
/*
  Prints a message on the Telnet and gdb console via the 'O' packet.
  Uses registers as follows:
  D0 = 0xaba00001       * trigger value *
  D1 = length of string
  A0 = address of string to print
*/

void bdi2000_console_out (const char* msg)
{
  register unsigned long _len __asm__("%d1") = strlen(msg);
  register const void* _msg __asm__("%a0") = msg;
  register const unsigned long _callid __asm__("%d0") = 0xaba00001UL;
  __asm__ __volatile__ ("bgnd");
}

int main(void)
{
  int  i;
  char message[32];

  while (1) {
    counter = 0;
    for (i = 0; i <= 16; i++) {
      counter++;
      fiboCount = Fibonacci(counter);
      sprintf(message, "\rFiboCount is %i\r\n", fiboCount);
      bdi2000_console_out (message);
    }
  }
  return 0;
}
```

In the GDB console you should then see:

```
(gdb) b main
Breakpoint 1 at 0x35fc: file fibo.c, line 65.
(gdb) c
Continuing.

Breakpoint 1, main () at fibo.c:65
65          counter = 0;
(gdb) c
Continuing.
FiboCount is 1
FiboCount is 1
FiboCount is 2
...
FiboCount is 610
FiboCount is 987
FiboCount is 1597

Breakpoint 1, main () at fibo.c:65
65          counter = 0;
(gdb)
```

3.4 Telnet interface

A Telnet server is integrated within the BDI. The Telnet channel is used by the BDI to output error messages and other information. Also some basic debug commands can be executed.

Telnet Debug features:

- Display and modify memory locations
- Display and modify general and special purpose registers
- Single step a code sequence
- Set hardware breakpoints (for code and data accesses)
- Load a code file from any host
- Start / Stop program execution
- Programming and Erasing Flash memory

During debugging with GDB, the Telnet is mainly used to reboot the target (generate a hardware reset and reload the application code). It may be also useful during the first installation of the bdiGDB system or in case of special debug needs (e.g. setting breakpoints on variable access).

Example of a Telnet session:

```
BDI>reset
- TARGET: processing user reset request
- TARGET: processing target init list ....
- TARGET: processing target init list passed

BDI>info
    Target state      : freezed
    Debug entry cause : startup break
    Current PC        : 0x00000040
BDI>
BDI>rd
D0 : ffffffff 040c1aa5 0dee4603 32821b4c
D4 : ffffffff ffffffff ffffffff ffffffff
A0 : ffffffff 23062afc ffffffff ffffffff
A4 : ffffffff 65223c76 ffffffff 0003fffe
PC : 00000040     SR : 0000271f
USP: d0c4ac41     SSP: 0003fffe
SFC: 00000005     DFC: 00000005
VBR: 00000000
BDI>
BDI>md 0x80000
00080000 : 0003fffe 00000040 ffffffff ffffffff  .......@........
00080010 : 42444931 30303020 4c445200 00000065  BDI1000 LDR....e
....
000800d0 : 4e740008 4e56fffc 2f074247 203caa55  Nt..NV../.BG <.U
000800e0 : 55aab0b9 000a0000 66202f39 000a0004  U.......f /9....
000800f0 : 2f39000a 00084eb9 000800ac b079000a  /9....N......y..
BDI>
```

Note:
The Telnet command RESET does only reset the target system. The configuration file is not loaded again. If the configuration file has changed, use the Telnet command BOOT to reload it.

Following a list of the available Telnet commands:

```
"MD    [<address>] [<count>]  display target memory as word (32bit)",
"MDH   [<address>] [<count>]  display target memory as half word (16bit)",
"MDB   [<address>] [<count>]  display target memory as byte (8bit)",
"DUMP  <addr> <size> [<file>] dump target memory to a file",
"MM    <addr> <value> [<cnt>] modify word(s) (32bit) in target memory",
"MMH   <addr> <value> [<cnt>] modify half word(s) (16bit) in target memory",
"MMB   <addr> <value> [<cnt>] modify byte(s) (8bit) in target memory",
"MC    [<address>] [<count>]  calculates a checksum over a memory range",
"MV                           verifies the last calculated checksum",
"RD    [<name>]               display CPU or user defined register",
"RDUMP [<file>]               dump all user defined register to a file",
"RM    <name> <value>         modify CPU or user defined register",
"RESET                        reset the target system",
"GO    [<pc>]                 set PC and start target system",
"TI    [<pc>]                 trace on instuction (single step)",
"TC    [<pc>]                 trace on change of flow",
"HALT                         force target to enter debug mode",
"BI  <addr> [2K|8K|32K]       set instruction breakpoint",
"BD  [R|W] <addr>             set data breakpoint (32bit access)",
"BDH [R|W] <addr>             set data breakpoint (16bit access)",
"BDB [R|W] <addr>             set data breakpoint ( 8bit access)",
"BDR [R|W] <addr> [2K|8K|32K] set data breakpoint on a range",
"CB                           clear data/instruction breakpoint",
"INFO                         display information about the current state",
"LOAD   [<offset>] [<file> [<format>]] load program file to target memory",
"VERIFY [<offset>] [<file> [<format>]] verify a program file to target memory",
"PROG   [<offset>] [<file> [<format>]] program flash memory",
"                            <format>  : SREC, BIN, AOUT, ELF or COFF",
"ERASE  [<address> [<mode>]]  erase a flash memory sector, chip or block",
"                    <mode>  : CHIP, BLOCK or SECTOR (default is sector)",
"DELAY  <ms>                  delay for a number of milliseconds",
"HOST   <ip>                  change IP address of program file host",
"PROMPT <string>              defines a new prompt string",
"CONFIG <file> [<hostIP> [<bdiIP> [<gateway> [<mask>]]]]",
"HELP                         display command list",
"BOOT  [loader]               reboot the BDI and reload the configuration",
"QUIT                         terminate the Telnet session"
```

Notes:
The DUMP command uses TFTP to write a binary image to a host file. Writing via TFTP on a Linux/ Unix system is only possible if the file already exists and has public write access. Use "man tftpd" to get more information about the TFTP server on your host.

4 Specifications

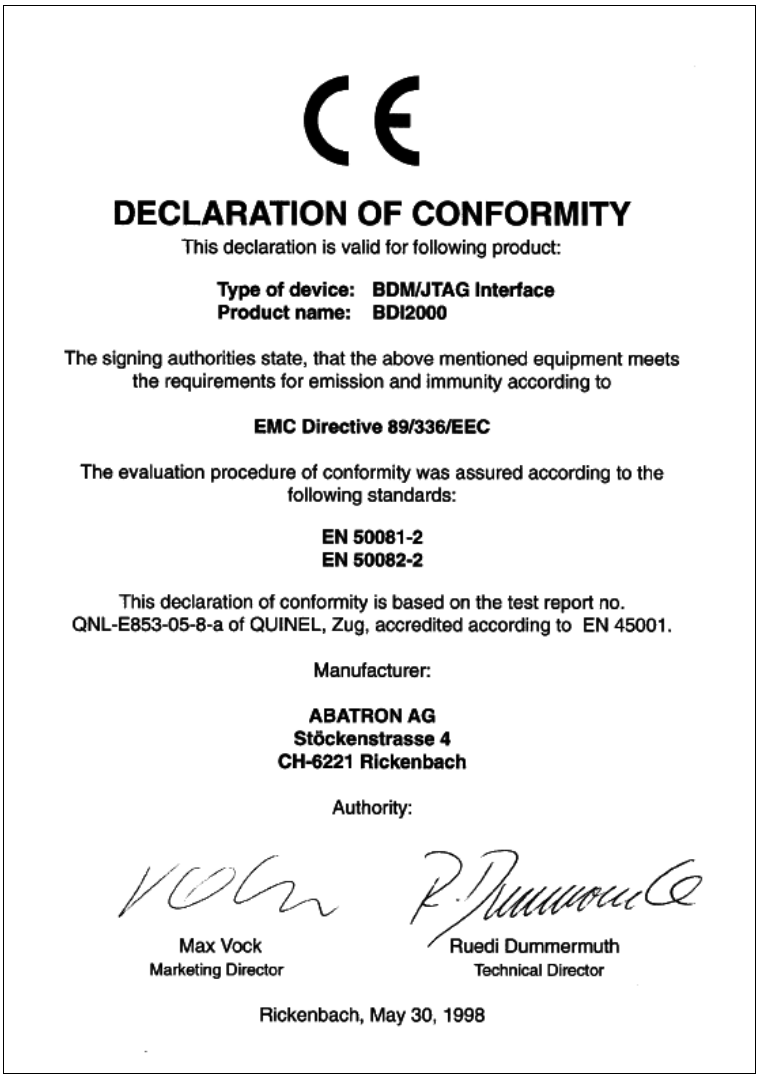
Operating Voltage Limiting	5 VDC ± 0.25 V
Power Supply Current	typ. 500 mA max. 1000 mA
RS232 Interface: Baud Rates	9’600,19’200, 38’400, 57’600,115’200
Data Bits	8
Parity Bits	none
Stop Bits	1
Network Interface	10 BASE-T
Serial Transfer Rate between BDI and Target	up to 16 Mbit/s
Supported target voltage	1.8 – 5.0 V (3.0 – 5.0 V with Rev. A/B)
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	190 x 110 x 35 mm
Weight (without cables)	420 g
Host Cable length (RS232)	2.5 m

Specifications subject to change without notice

5 Environmental notice

Disposal of the equipment must be carried out at a designated disposal site.

6 Declaration of Conformity (CE)



CE

DECLARATION OF CONFORMITY

This declaration is valid for following product:

Type of device: **BDM/JTAG Interface**
Product name: **BDI2000**

The signing authorities state, that the above mentioned equipment meets the requirements for emission and immunity according to

EMC Directive 89/336/EEC

The evaluation procedure of conformity was assured according to the following standards:

EN 50081-2
EN 50082-2

This declaration of conformity is based on the test report no. QNL-E853-05-8-a of QUINEL, Zug, accredited according to EN 45001.

Manufacturer:

ABATRON AG
Stöckenstrasse 4
CH-6221 Rickenbach

Authority:

[illegible] [illegible]

Max Vock
Marketing Director

Ruedi Dummermuth
Technical Director

Rickenbach, May 30, 1998

7 Abatron Warranty and Support Terms

7.1 Hardware

ABATRON Switzerland warrants that the Hardware shall be free from defects in material and workmanship for a period of 3 years following the date of purchase when used under normal conditions. Failure in handling which leads to defects or any self-made repair attempts are not covered under this warranty. In the event of notification within the warranty period of defects in material or workmanship, ABATRON will repair or replace the defective hardware. The customer must contact the distributor or Abatron for a RMA number prior to returning.

7.2 Software

License

Against payment of a license fee the client receives a usage license for this software product, which is not exclusive and cannot be transferred.

Copies

The client is entitled to make copies according to the number of licenses purchased. Copies exceeding this number are allowed for storage purposes as a replacement for defective storage mediums.

Update and Support

The agreement includes free software maintenance (update and support) for one year from date of purchase. After this period the client may purchase software maintenance for an additional year.

7.3 Warranty and Disclaimer

ABATRON AND ITS SUPPLIERS HEREBY DISCLAIMS AND EXCLUDES, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.

7.4 Limitation of Liability

IN NO EVENT SHALL ABATRON OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING, WITHOUT LIMITATION, ANY SPECIAL, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE HARDWARE AND/OR SOFTWARE, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, BUSINESS, DATA, GOODWILL, OR ANTICIPATED SAVINGS, EVEN IF ADVISED OF THE POSSIBILITY OF THOSE DAMAGES.

The hardware and software product with all its parts, copyrights and any other rights remain in possession of ABATRON. Any dispute, which may arise in connection with the present agreement shall be submitted to Swiss Law in the Court of Zug (Switzerland) to which both parties hereby assign competence.

Appendices

A Troubleshooting

Problem
The firmware can not be loaded.

Possible reasons

- The BDI is not correctly connected with the target system (see chapter 2).
- The power supply of the target system is switched off or not in operating range (4.75 VDC ... 5.25 VDC) --> MODE LED is OFF or RED
- The built in fuse is damaged --> MODE LED is OFF
- The BDI is not correctly connected with the Host (see chapter 2).
- A wrong communication port (Com 1...Com 4) is selected.

Problem
No working with the target system (loading firmware is ok).

Possible reasons

- Wrong pin assignment (BDM/JTAG connector) of the target system (see chapter 2).
- Target system initialization is not correctly --> enter an appropriate target initialization list.
- An incorrect IP address was entered (BDI2000 configuration)
- BDM/JTAG signals from the target system are not correctly (short-circuit, break, ...).
- The target system is damaged.

Problem
Network processes do not function (loading the firmware was successful)

Possible reasons

- The BDI2000 is not connected or not correctly connected to the network (LAN cable or media converter)
- An incorrect IP address was entered (BDI2000 configuration)

B Maintenance

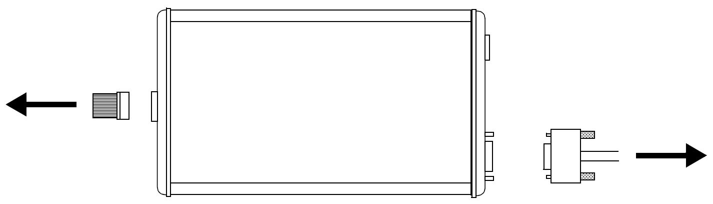
The BDI needs no special maintenance. Clean the housing with a mild detergent only. Solvents such as gasoline may damage it.

If the BDI is connected correctly and it is still not responding, then the built in fuse might be damaged (in cases where the device was used with wrong supply voltage or wrong polarity). To exchange the fuse or to perform special initialization, please proceed according to the following steps:

Observe precautions for handling (Electrostatic sensitive device)
Unplug the cables before opening the cover.
Use exact fuse replacement (Microfuse MSF 1.6 AF).

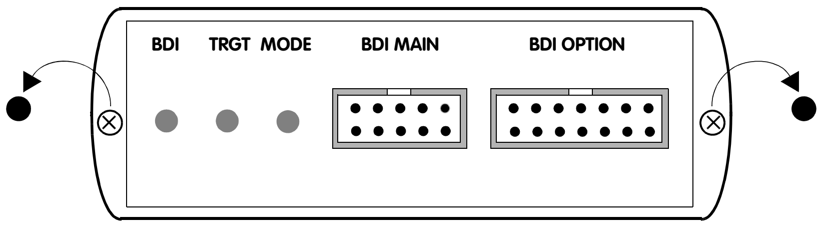
1

1.1 Unplug the cables



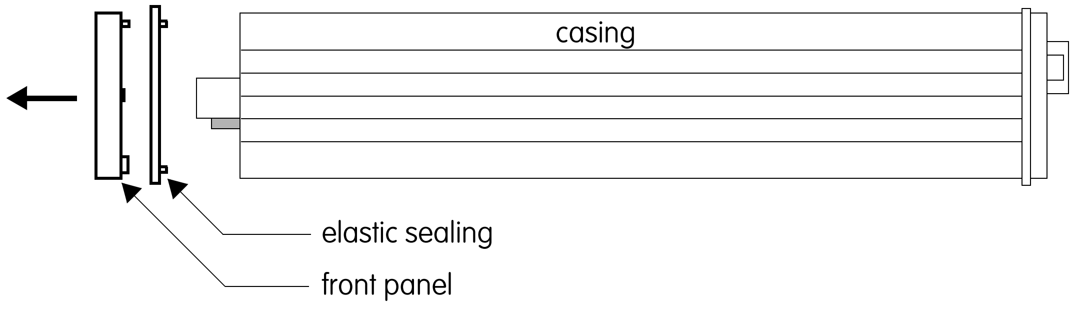
2

2.1 Remove the two plastic caps that cover the screws on target front side (e.g. with a small knife)
2.2 Remove the two screws that hold the front panel



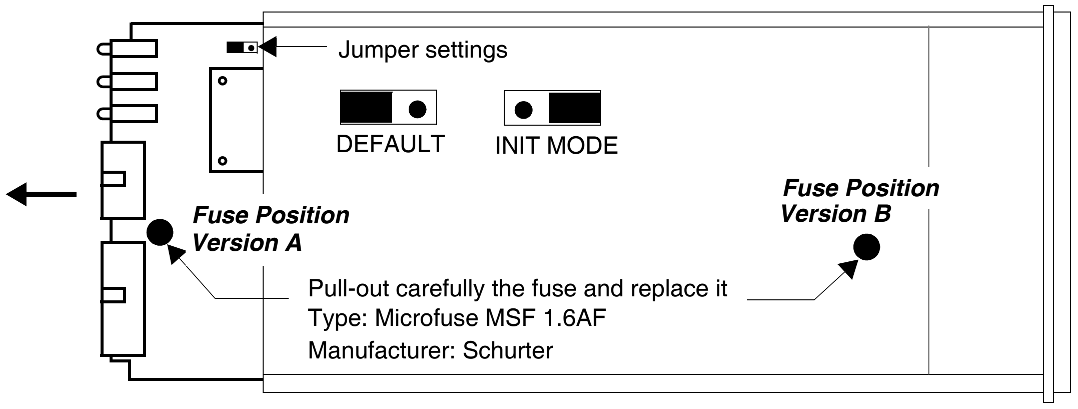
3

3.1 While holding the casing, remove the front panel and the red elastig sealing



4

4.1 While holding the casing, slide carefully the print in position as shown in figure below

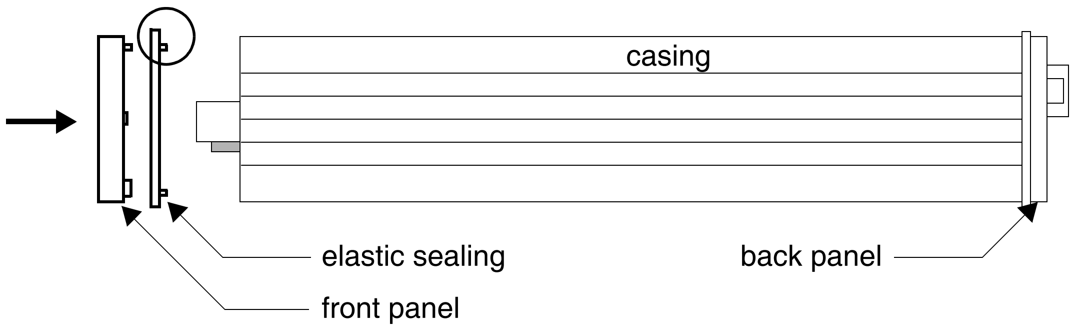


5

Reinstallation

5.1 Slide back carefully the print. Check that the LEDs align with the holes in the back panel.

5.2 Push carefully the front panel and the red elastig sealing on the casing. Check that the LEDs align with the holes in the front panel and that the position of the sealing is as shown in the figure below.



5.3 Mount the screws (do not overtighten it)

5.4 Mount the two plastic caps that cover the screws

5.5 Plug the cables

Observe precautions for handling (Electrostatic sensitive device)
Unplug the cables before opening the cover.
Use exact fuse replacement (Microfuse MSF 1.6 AF).

C Trademarks

All trademarks are property of their respective holders.